Use of Discrete Bayesian Classifiers for Music Recommendation

Jackson Firlik
University of Miami
Music Engineering
Coral Gables, FL
j.firlik@umiami.edu

Connor McCullough
University of Miami
Music Engineering, Electrical Engineering
Coral Gables, FL
c.mccullough1@umiami.edu

***Abstract*—This study is intended to determine the feasibility of applying a discrete Bayesian Classifier to musical artist metadata for artist recommendations to the end user. The study concludes that accurate classification is possible given a lack of noise in test data. A moderate degree of accuracy is achieved with a relatively small number of attributes and test samples, proving this algorithm is feasible for app-based music recommendations.**

Keywords—Bayes; music informatics; machine learning;

I. INTRODUCTION

Music Informatics is a growing field which deals with how music is consumed, distributed, and produced. Companies such as last.fm and Echonest provide the service of music recommendations based on the knowledge of a user's music taste, and the listening patterns of other users who listen to the same artists. However, recommendations are much more difficult without an established network of users and what they listen to. In theory, it should be possible to provide recommendations to a single user, given their music taste solely based on the metadata of an artist including their origin, instrumentation, albums, etc. The goal of this project is to determine if recommendations solely based on metadata are possible by examining the attributes of top listened artists of the two authors. A portion of the subjects' libraries will be used to generate training data, while the rest will be used to test if the classifier can properly determine which subject's library the artist belongs to. The project will be implemented using a Bayesian Classifier written in MATLAB, using 20 test examples each and 10 attributes. If the findings of this study are conclusive, the algorithm can be applied in many music recommendation based smart phone and PC applications.

II. BACKGROUND

A. Bayesian Classifiers

The Bayesian Classifier, is one of the most basic, yet accurate Machine Learning Algorithms. Like other classification algorithms, the Bayesian Classifier uses the attributes and classes of training data to determine the class of test data based on its attributes. It utilizes basic Probability concepts such as a priori probability and conditional probability, and is calculated using Bayes' Theorem, defined by:

$$P(y|x) = \frac{P(x,y)P(y)}{P(x)} \tag{1}$$

The notation $P(x|y)$ signifies conditional probability, which means the probability that x will occur based on the information that y has already occured. In the Bayesian Classifier, y is replaced by the class being calculated and x is replaced by a particular attribute:

$$P(c_i|x) = \frac{P(x|c_i)P(c_i)}{P(x)} \tag{2}$$

Based on this equation, it can be seen we are trying to calculate the probability of a particular classification occurring, based on the occurrence of a particular attribute. This can be done for any number of attributes, so the class with the highest sum of probabilities at the end is what the classifier chooses:

$$P(c_i) = \sum_{j=1}^{r} \frac{P(x_j|c_i)P(c_i)}{P(x_j)} \tag{3}$$

As seen in equations 1-3, in order to return a classification, the conditional probability of a class given an attribute $P(c_i|x)$, the probability of each class occurring on its own $P(c_i)$, and the probability of each attribute occurring on its own $P(x)$. Because $P(x)$ will be the same for each class, and therefore doesn't affect the classification outcome, it can be ignored from the equation, making the practical Bayesian classifier formula:

$$P(c_i) = \sum_{j=1}^{r} P(x_j|c_i)P(c_i) \tag{4}$$

B. Attribute Selection

For the application of this project, the two classes were chosen to be taste of author 1, and taste of author 2. 11 Attributes were created which were determined to be potential deciding factors in determining taste in musical taste. At its most basic, musical taste is determined by the listeners perception of each component in the music. Age of frontman was chosen because the age of the main writer determines what music they listened to while developing, and music is distinctly different throughout time. For this reason, the year the band was established was also included. The number of members is included because more members could mean a more intricate and layered sound, while less members may mean a more straightforward and simple sound. The presence of each

instrument is important because some listeners may enjoy different instruments to different degrees than others. Facebook fans is an attribute because it is an important indicator as to the popularity of the band, which determines whether a listener likes more "mainstream" music, or "underground music". Time between albums was included because it shows how long a band took to make each album. Different styles of music take different amounts of time to produce, with some bands putting out albums every year and others taking many years.

All the attributes in this project are discrete, however some such as age, year, and members contain a large number of potential values. One fundamental principle of probability is that as the number of possible discrete values increases, the probability of each value occurring gets less and less, eventually approaching zero. This means that although some values such as number of fans are discrete values, the number of possible values makes them virtually continuous. Because continuous and discrete Bayesian Classifiers are completely different algorithms, all attributes must either be continuous or discrete. Many of the attributes with a large number of possible values were discretized to avoid overloading the algorithm with two many possible values, and creating rounding error with extremely small probabilities. Discretizing is a simple process consisting of rounding values and placing them in a certain number of "bins". For simplicities sake, all of the rounding is linear, meaning each of the bins has an equally large range.

III. IMPLEMENTATION

A. Structure

Our implementation in Matlab was fairly straightforward. In our main script, we read in our training set data from a spreadsheet as a 2-dimensional vector, and separated it by their given classifiers into two separate vectors. We then wrote a function called *calcProbs* to do the initial probability calculations. For each particular attribute vector, we found the probability that each individual attribute value would occur and placed these individual probabilities in a new vector. Each of these vectors (one for each attribute) was then placed in a cell array to be returned from the function. With the probabilities for each attribute calculated, we then wrote a function called *calcBayes* to perform the Bayesian equation on the data. For this function, we input new data to test for classification, along with the previously calculated probabilities for each individual class. Although the Bayesian formula technically also requires the total probability for each attribute, this value is the same for each class and therefore only serves to scale the final probability; therefore it can be neglected. With the Bayesian formula applied, classification requires simply choosing the class with the higher probability.

B. Testing

To test (1) the accuracy of the algorithm and (2) its ability to perform classifications based on real training/test data, two different tests were performed with two sets of test data. The first test was done using artificially generated training/test data, which had huge gaps in the attribute values for each class, so that the chosen class should be extremely obvious.

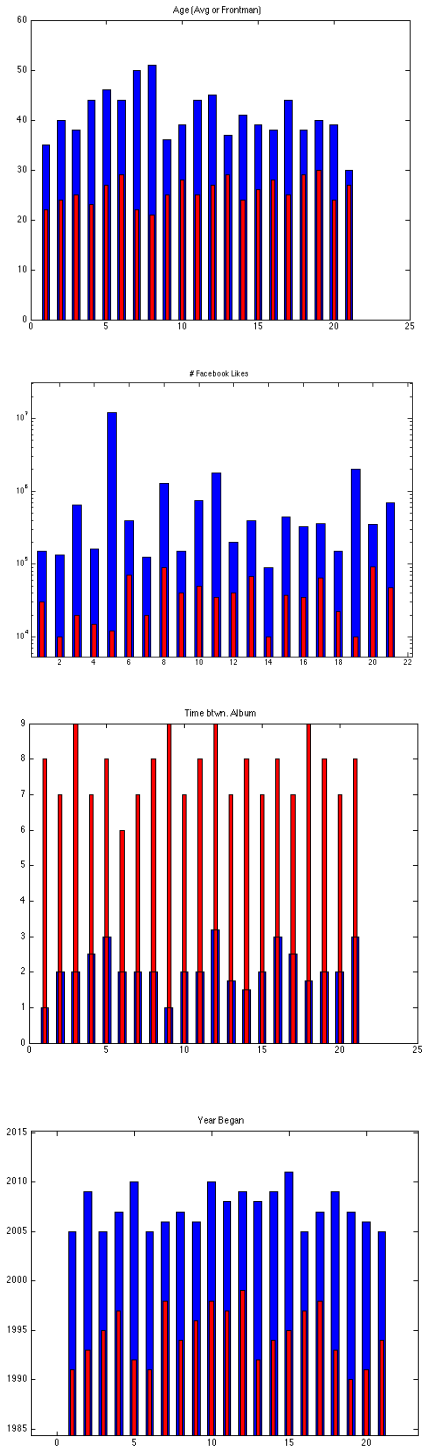

Fig. 1. Comparison of artificially generated attribute values for both classes for attributes (in descending order) Age, Facebook likes, Length of time between albums, and Year started.

Using this artificial data, the algorithm classified perfectly every time, showing that the code was indeed performing

Bayesian Classification. The second set of data was actual values for each artist from the subjects' libraries. Upon visual inspection of the training data, it can be seen that the attributes values for both classes are much closer together.

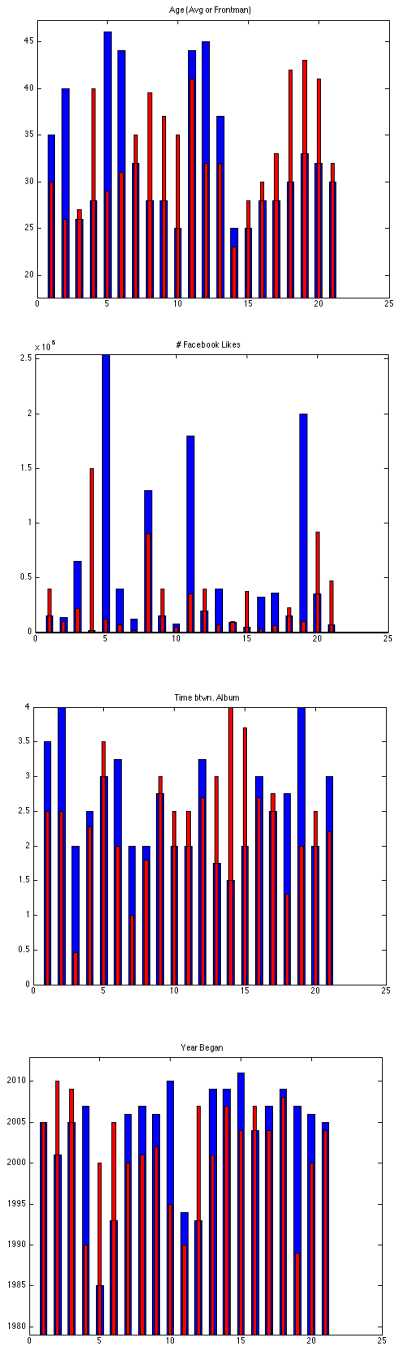

Fig. 2. Comparison of real attribute values for both classes for attributes (in descending order) Age, Facebook Likes, Length of time between albums, and Year started.

Despite this observation, it is possible that the algorithm would be able to detect small differences that were unseen by visual inspection. However, the test results gave a much lower accuracy, showing that a classification could not be made consistently based on the given training examples.

IV. Results & Problems

It is possible that our experiment was greatly hindered by a lack of attributes, or at least a lack of relevant, important attributes. There are many other factors that determine an artist's sound or style that cannot be quantified with age, instrumentation, popularity or other easily observable attributes. However, there may yet be attributes that are easily observable which would improve the performance of this algorithm. For example, total classification of all possible instrumentations, not limited to guitars, keyboards, horns, and drums, would most likely improve classification accuracy; this data may be too cumbersome to collect by hand but could easily be read from a database if one existed. More data about the music itself would also improve the performance, such as overall song or album length which could easily be calculated from a database. Apart from this metadata about the music, actual data about the content of the music itself would greatly enhance classification accuracy, at the expense of using more complicated signal processing throughout the algorithm. Data about frequency content, tempo and beat-mapping would tell a lot about the style of the music. While these algorithms are outside the scope of this experiment, they would certainly be useful in dealing with the objective classification of musical artists.

In an ideal situation, an infinite number of attributes would be used to classify the musical data with utmost precision, however there are problems with this. As we encountered when collecting our data, certain attributes play a more important role in classification than others. The number of Facebook Likes an artist has may be indicative of general trends in style (>1,000,000 likes probably indicates a more pop-oriented artist), however on smaller scales (<100,000 likes) says nothing about the sound of the artist. A drawback of using Bayesian classifiers is the lack of a weighting system for the attributes; with all attributes weighted equally, variance within less important attributes can mislead the classifiers. A possible improvement to the algorithm could be calculating the class probabilities with a variety of different weighting functions and seeing which gives the greatest amount of accuracy. This would be desirable because the process of guessing at or tweaking weights until results are desirable is automated, and can create configurations that are not intuitive to the human eye. It is also desirable because many algorithms have hundreds of attributes, making any manual setting of weighting coefficients cumbersome. In some cases, attributes may be completely unhelpful in generating an accurate classification. Discarding these can increase accuracy by eliminating noise, as well as increasing the processing speed of the algorithm.

The attribute of Facebook Likes brings to light another issue: some of our attributes (age, Facebook Likes, time between albums) began as essentially continuous attributes which we then discretized using linear discretization. Whether or not linear discretization is the appropriate method is the question, as for a case like Facebook Likes it appears that a logarithmic scheme may work better. The difference from 10,000 likes to 100,000 likes is much greater than the difference between 1,000,000 and 1,100,000. Utilizing a logarithmic discretization method may help improve the

relevance of this data. While it is easy to analyze this particular attribute and see the necessity of logarithmic discretization, with a large number of attributes this may be difficult. Similar to automating setting of weighting coefficients, automated setting of discretization could also benefit the overall accuracy of the program. Performing something similar to histogram equalization in image processing may be desirable, as the ideal discretization pattern is to have an equal number of values in each bin.

V. Conclusion

With ideal training and testing information, the algorithm performed extremely accurately. However, with practical data from the test subjects, the algorithm performed much more poorly and was often unable to classify accurately. Despite poor performance on practical data, implementing the aforementioned improvements could make the algorithm more robust, allowing this system of classification to be utilized in musical recommendation applications of greater scope.